Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Cancelled).

2. (Currently amended) The processor of claim 11, wherein the critical excepted instructions comprise exceptions that are performance critical.

3. (Currently amended) The processor of claim 11, wherein the non-critical excepted instructions comprise exceptions that are not performance critical.

4. (Original) The processor of claim 2, wherein the critical excepted instructions include branch mispredictions.

5. (Original) The processor of claim 2, wherein the critical excepted instructions include load/store traps.

6. (Original) The processor of claim 2, wherein the critical excepted instructions include jump mispredictions.

7. (Original) The processor of claim 3, wherein the non-critical excepted instructions include illegal instructions.

8. (Original) The processor of claim 3, wherein the non-critical excepted instructions include cache parity errors.

9. (Original) The processor of claim 3, wherein the non-critical excepted instructions include invalid instructions.

10. (Original) The processor of claim 3, wherein the excepted instructions include arithmetic overflows.

11. (Previously presented) A processor, comprising:

a first exception handler that receives and handles critical excepted instructions, wherein the first exception handler handles the critical excepted instructions even if the critical excepted instructions are still speculative; and

a second exception handler that receives and handles non-critical excepted instructions.

12. (Previously presented) A processor, comprising:

a first exception handler that receives and handles critical excepted instructions, wherein the first exception handler causes critical excepted instructions to be resolved on a speculative basis, even though the excepted instruction may not be in an actual program path; and

a second exception handler that receives and handles non-critical excepted instructions.

13. (Currently amended) The processor of claim 11, wherein the second exception handler operates non-speculatively.

14. (Currently amended) The processor of claim 11, wherein the second exception handler causes non-critical excepted instructions to be resolved only when it is certain that the excepted instruction is in an executing program.

15. (Currently amended) The processor of claim 11, further comprising a plurality of pipelines with multiple stages, and wherein excepted instructions may arise in one or more of the pipeline stages.

16. (Original) The processor of claim 15, wherein excepted instructions arising from said one or more pipeline stages is routed to the first exception handler or second exception handler based on a predetermined criteria.

17. (Original) The processor as in claim 16, wherein the predetermined performance criteria relates to performance of the processor.

18. (Cancelled).

19. (Previously presented) An exception handler for a processor that resolves excepted instructions, comprising:

a speculative exception handler that receives critical excepted instructions and resolves said critical excepted instructions on a speculative basis, wherein the speculative exception handler causes critical excepted instructions to be expeditiously resolved even though the critical excepted instruction may not be in an actual path of an executing program; and

a non-speculative exception handler that receives non-critical excepted instructions and resolves said non-critical excepted instructions on a non-speculative basis.

20.-21. (Cancelled).

22. (Previously presented) A processor, comprising:

at least one pipeline with a plurality of stages;

an algorithm for detecting non-executable instructions in said at least one pipeline, wherein said algorithm generates a command that identifies the non-executable instruction and identifies a reason that the non-executable instruction will not execute;

a speculative exception handler that receives said command for any non-executable instructions that are critical to processor performance,

wherein the speculative exception handler expeditiously resolves critical non-executable instructions even though the critical non-executable instruction may not be in an actual path of an executing program; and

a non-speculative exception handler that receives said command for any non-executable instructions that are not critical to processor performance.

23. (Cancelled).

24. (Original) The processor as in claim 22, wherein said speculative exception handler includes logic for resolving critical non-executable instructions.

25.-26. (Cancelled).

27. (Previously presented) A method of handling exceptions in a processor during the execution of a program, comprising:

detecting an exception in one or more stages of one or more pipelines;

identifying if the exception is critical to the performance of the processor;

routing critical exceptions to a first exception handler;

routing all non-critical exceptions to a second exception handler;

resolving critical exceptions by handling the critical exceptions even if the instructions associated with the critical exceptions are still speculative.

28. (Previously presented) A method of handling exceptions in a processor during the execution of a program, comprising:

detecting an exception in one or more stages of one or more pipelines;

identifying if the exception is critical to the performance of the processor;

routing critical exceptions to a first exception handler;

routing all non-critical exceptions to a second exception handler; and

expeditiously resolving critical exceptions even though the critical exception may not be in an actual path of the program.

29.-31. (Cancelled).

32. (Previously presented) A method of handling exceptions in a processor during the execution of a program, comprising:

detecting an exception and identifying if the exception is critical or non-critical to the processor performance;

routing critical exceptions to a speculative exception handler;

routing all non-critical exceptions to a non-speculative exception handler; and

resolving the critical exceptions even though the critical exceptions may not be in an actual path of the program.

33.-34. (Cancelled).

35. (Previously presented) A method of handling exceptions in a processor during the execution of a program, comprising:

detecting an exception in one or more stages of one or more pipelines;

determining whether the exception is critical to the performance of the processor; and

resolving critical exceptions even though the critical exceptions may not be in an actual path of the program.

36. (Previously presented) The method as defined in claim 35 further comprising:

routing critical exceptions to a first exception handler; and

routing all non-critical exceptions to a second exception handler.

37. (Previously presented) An exception handler that resolves excepted instructions, comprising:

a speculative exception handler that receives critical excepted instructions and resolves said critical excepted instructions on a speculative basis, wherein the speculative exception handler causes critical excepted instructions to be expeditiously resolved even though the critical excepted instruction may not be in an actual path of an executing program; and

a non-speculative exception handler that receives non-critical excepted instructions and resolves said non-critical excepted instructions on a non-speculative basis.